E0-271 Final Project

A Real-Time Framework for Eddy Visualization and Analysis

Tanay Narshana, Tushar Pandurang Kadam

Problem Statement

Oceanic Eddies have been of great interest to oceanographers, weather analysts, and many other domain experts for their ability to transport energy, in various forms, across the globe. The problem is that these domain experts cannot directly infer eddies and their properties from the large amount of data the satellites fetch. It is also difficult to track the real-time status of eddies in oceans due to the requirement of processing large-scale data from the satellites efficiently. In this regard, as students of the visualization community, we aim to aid these domain experts with an approach to visualize and get information about the eddies in real-time.

Related Work

- Sedat Ozer et al.[1] derive eddies from the Okubo-Weiss[2, 3] parameter. They then work on the 3D domain and use Computer Vision techniques on the eddies in a 3D domain for tracking. Computations on a 3D domain involve higher time complexities than computations on a 2D domain (a surface of the 3D domain). They also use a volume overlap technique to track the eddies over time. In our opinion, this technique is bound to fail from our observation of the eddies(more on this below).
- The Okubo-Weiss parameter is shows a high number of false detections due to noise in the field of the parameter. Dujuan Kang et al.[4] show that the Okubo-Weiss parameter, a 2D metric applied to the sea surface with additional geometric constraints, can detect eddies successfully.

Work Completed

Our real-time framework's core idea is based on attaining faster processing and data-transfers across various stages of our visualization pipeline. To achieve this, we convert the satellite data to a 3D bit pattern (which has 1s when a corresponding point was within an eddy and 0s otherwise) as earliest as possible. We develop scripts for analysis of eddies at a given instance in time, such as the number of eddies, radii of each eddy, etc. We also develop a simple yet elegant approximation algorithm to track eddy activities such as birth, death, split, and merges of eddies based on an oceanographic insight that eddy transformations are not abrupt. This algorithm, hence, can aid us in tracking the eddies as well. We also discuss the reason for the failure of this algorithm when applied on the given data set[1][5]. Finally, we use image processing to detect and track eddy birth and death overtime for the given dataset.

Tasks Done

- We use the very popular, Okubo-Weiss parameter, a 2D measure of vorticity, to detect our eddies. The parameter is prone to a huge rate of false detections. With the help of heuristics suggested by Dujuan Kang et al.[4] and certain modifications to account for noise, we could detect eddy centers as shown in red in *Figure* 1.

[1]Red Sea data courtesy of Red Sea Modeling and Prediction Group (PI Prof. Ibrahim Hoteit), KAUST available at https://kaust-vislab.github.io/SciVis2020/data.html

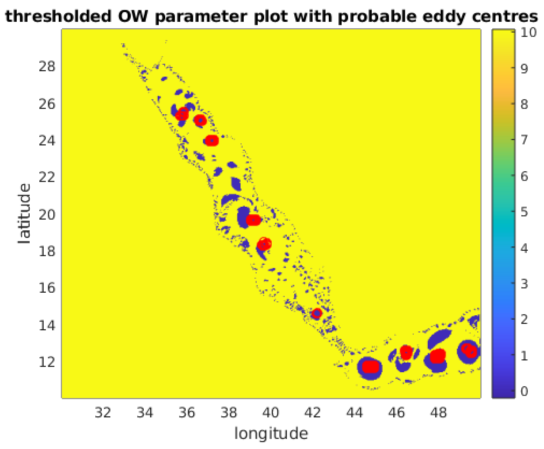
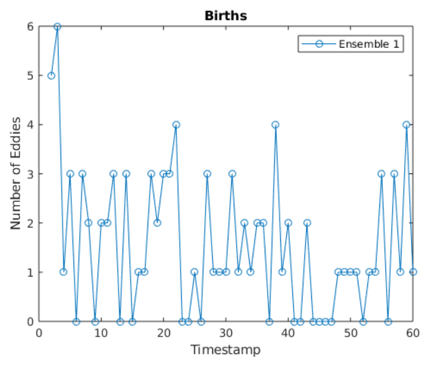


Figure 1: Eddy centres found (in red) in Ensemble 1 for Timestamp 15

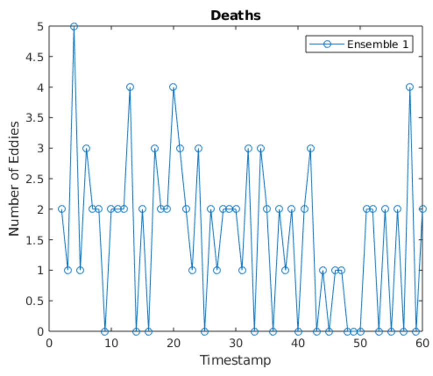
Figure 1 shows the eddies by the purple-colored entities as per the Okubo-Weiss (or OW) parameter. We then construct eddies in 3D from these centers by applying breadth-first-search from each of the centers until the OW parameter is in the desired limit. Since we find eddies using the OW parameter, which is also a 2D metric by nature, on the sea surface flow, we feel that it would not be appropriate to use a 3D model of any sort to gain inferences about the eddies. This feeling reinforces our idea of using a 2D space (the sea surface) to analyze the eddies. Hence we choose to do this.
The eddies found in 3D are then stored into a netCDF file, which contains 0s and 1s. Since Inviwo (the tool we use for visualizing eddy in 3D over time) does not support boolean variables, we store the variables in uint8 format. Had we stored it in boolean format, each timestep for any ensemble would have had a size less than 2MB!

- We only use the bit pattern generated in the previous task for the sea surface to analyze eddies. In this task, we study the number of eddies, the effective radii of each eddy and track eddies' evolution over time. This evolution of the eddy plot helps us infer the death, merge, split, and the birth of eddies. Initially, we designed an algorithm (described in the following subsection) to track eddy events (birth, death, split, merge) using an oceanographic insight that the eddy transformations are not abrupt. This algorithm (and also the volume overlap approach used by Sedat Ozer et al.[1]) fails because of the large time difference (12 hours) between two consecutive time steps in the given data set. Between two consecutive time step samples of the Red Sea data, one can expect changes that are quite significant, and we observe this while manually going through the extracted eddies.

  On applying our novel approximation algorithm to ensemble 1, we get the plots for Births and Deaths of eddies as shown in *Figure* 2.



(a) Number of Births for each time step



(b) Number of Deaths for each time step

Figure 2: Number of births and deaths for each time step in Ensemble 1

While the plots in *Figure* 2 may look convincing, a manual search for the truth in the given data set shows that the plot is a bluff of what is actually happening.
So, to actually capture events and track eddies for the given data set, we would require more sophisticated techniques from *Image Processing* and *Computer Vision*. That is what we exactly do.

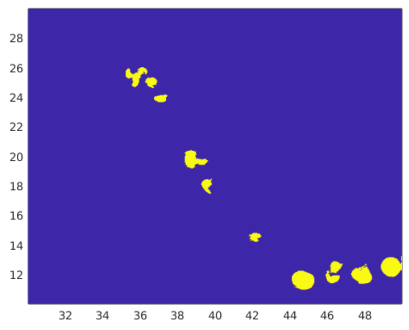
Eddy Activity Detection Algorithm for Real-Time Analysis

Input: I_t, I_{t+1} (two binary images from consecutive time steps of the sea surface where eddy points have value 1 and vice-versa)

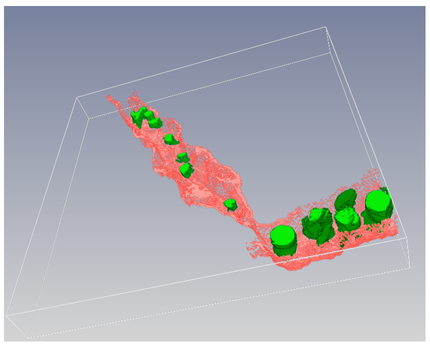
```
1:  Extract connected components of 1s from both images
2:  while each connected component of $I_t$ is not visited do
3:      $C_i$ = a connected component of $I_t$ not yet visited.
4:      $S_{t+1}$ = set of all connected components in $I_{t+1}$ that overlaps $C_i$.
5:      $S_t$ = set of all connected components in $I_t$ that overlaps any $C_j \in S_{t+1}$.
6:      if $|S_t| == |S_{t+1}| == 1$ then
7:          No Event
8:      else if $|S_t| == 1$ AND $|S_{t+1}| > 1$ then
9:          Split
10:     else if $|S_t| > 1$ AND $|S_{t+1}| == 1$ then
11:         Merge
12:     else if $|S_{t+1}| == 0$ then
13:         Death
14:     else
15:         Complex Event
16:     end if
17:     Mark all components in $S_t$ and $S_{t+1}$ as visited
18: end while
19: For all unvisited connected components of $I_{t+1}$ report Birth.
20: return $T$
```

Results and Evaluation

1. In *Task 1* that we proposed, we obtain the visualization of eddies as shown in *Figure* 3.



(a) Bird's eye view



(b) 3D View

Figure 3: Visualization of Eddies for Time Step 15 in Ensemble 1

We will show the time-varying 3D Visualization of the eddies during the presentation.

2. In *Task 2* that we proposed, we first find the number of eddies for all time steps across ensembles 1 to 7 as shown in *Figure* 4 by finding the number of connected components for each time step.

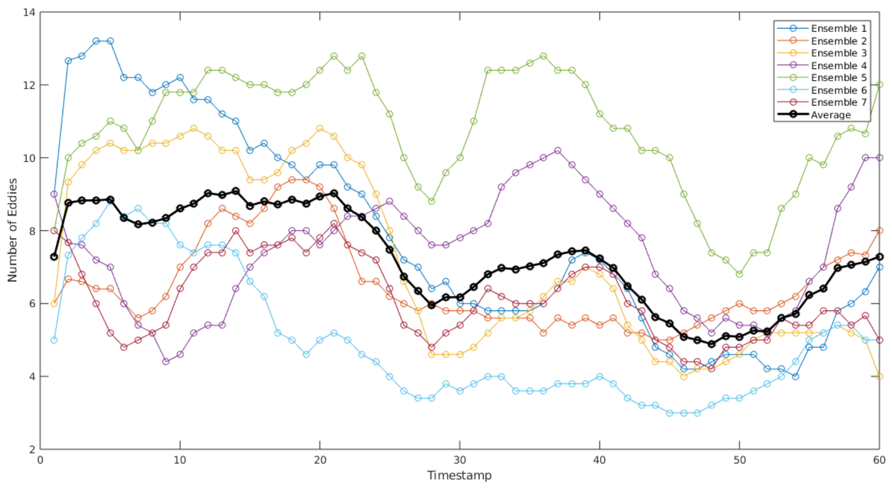


Figure 4: Number of Eddies for ensembles 1 to 7

To find the Effective Radii of each eddy for a particular time step, we use the *Pick's Theorem*. An Effective Radii plot is shown in *Figure* 5.

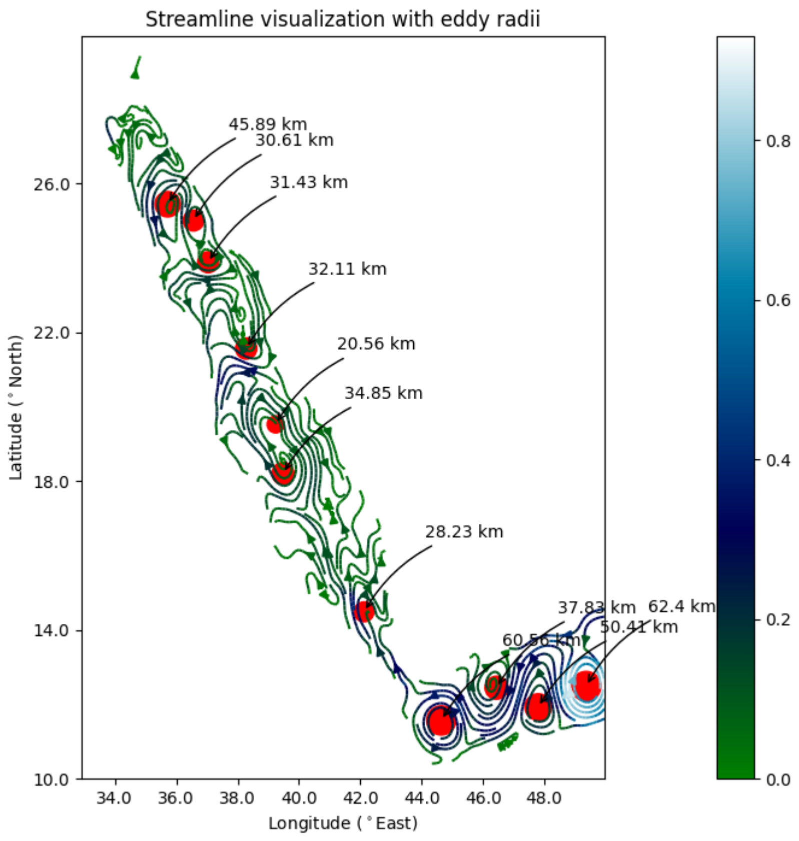


Figure 5: Effective Radii of Eddies in Ensemble 1 for Timestamp 15

Finally, we obtain a plot depicting the evolution of the eddies over time using OpenCV. This plot track eddies over time. One such evolution plot, for Ensemble 1, is shown in *Figure* 6.

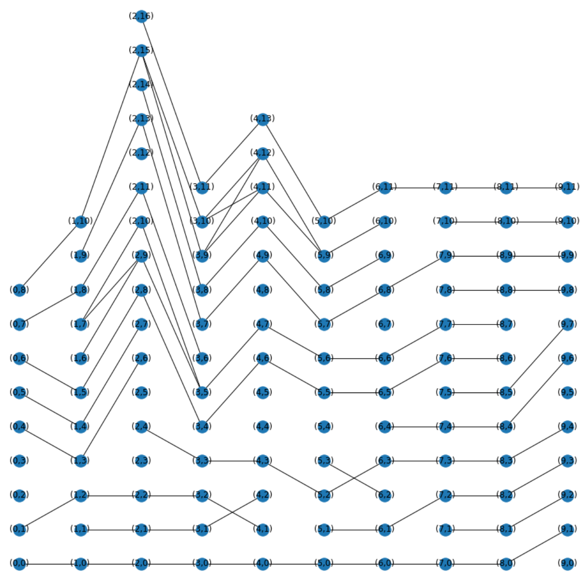
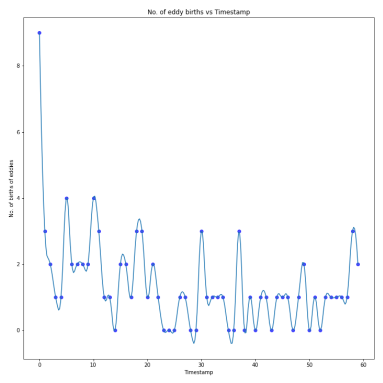
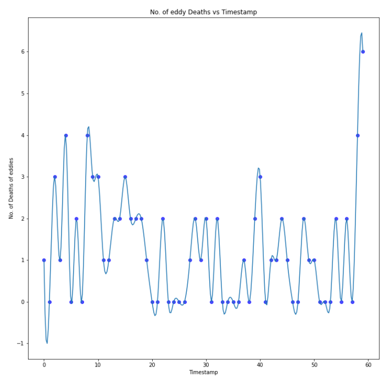


Figure 6: Evolution Plot for Eddies of Ensemble 1 [legend: (x,y) = (timestep,eddy index)]

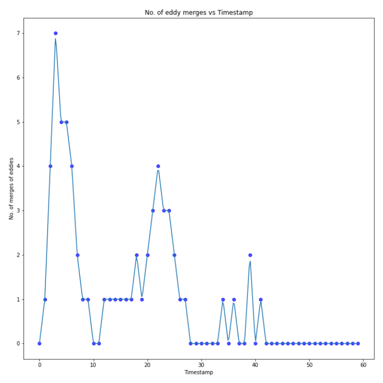
From this plot, we were able to derive birth, death, merge and splits in eddies over time as shown in *Figure* 7 for Ensemble 1.



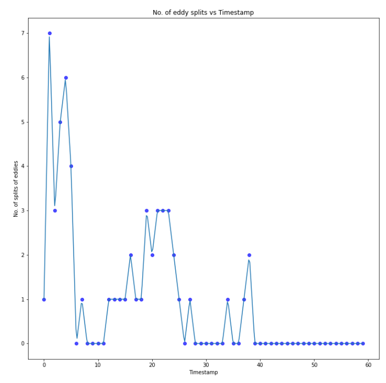
(a) Eddy Births over time



(b) Eddy Deaths over time



(c) Eddy Merges over time



(d) Eddy Splits over time

Figure 7: Eddy Activity for Ensemble 1

Though this approach of ours is not 100% accurate, it has a much better accuracy of detecting eddy activities than the earlier approach.

3. *Performance of the Framework*: The most time consuming script, among the ones that we write, is that of eddy detection using the OW parameter. It takes about 3.8 seconds in the worst case to detect eddies for each time step on our old, basic laptops. This is primarily because the domain of operation for eddy detection is 3D in nature. Following this, the scripts used for analysis easily execute within one second because their domain of operation is in 2D.
The performance of the eddy detection script can be easily brought down under 1 second using better hardware. Post this stage, the amount of data required for analysis is less than 2MB per timestep. Hence, we can efficiently transfer the data to the scripts performing analysis in a small fraction of a second. Hence, in a pipelined framework, the rate of updation of the framework would be the maximum of the eddy detection script and the time to fetch and preprocess data(in a pipelined fashion) from the satellites.

Discussion

The most difficult feature to develop, undoubtedly, was eddy tracking. Having less expertise in the domain of Image Processing, we struggled a lot to find appropriate techniques to identify similar blobs (blobs are eddies in our context) across consecutive time steps. Being stuck between research papers, probably borrowing some ideas from each of them, we were able to write our own scripts for the task.
The most exciting feature to develop was Eddy Centre detection. This was primarily because on following the steps provided by Kang et al. [4], we weren't able to see the eddies from a noisy OW parameter plot. We then tried our best to understand what each step in their paper meant and eventually made certain modifications involving a lot of smart image filters (to use MATLAB's optimised code) to achieve our goals.

Responsibilities of Each Member

As a team, we employed a hybrid-pair-programming approach towards the execution of the project. Since we come from different backgrounds, we read select papers individually and discuss them to understand the same paper from different focuses better. This enhanced our overall understanding.
Tushar feels comfortable with Python, and Tanay feels comfortable with MATLAB. This enabled us to simultaneously work on the same task on two different tools for a short period of time and then migrate to the more convenient tool to write the script for each task. In this way, we spent almost no time debugging since we could catch bugs fleetingly.
If we were to break down each member's contributions, it could only be segregated by the tools used to implement a particular task. Tanay primarily implemented the eddy detection part (proposed Task 1), and Tushar primarily implemented the eddy analysis part (proposed Task 2).

References

[1] S. Ozer, K. Bemisa, W. Hua, A. Goktogan, M. Aydogan, K. Guo, D. Kang, and L. L. nad Deborah Silver, "The Use of 3D Optical Flow, Feature-Tracking and Token-Tracking Petri Nets to Analyze and Visualize Multiple Scales of Ocean Eddies," 2020.

[2] A. Okubo, "Horizontal dispersion of floatable particles in the vicinity of velocity singularities such as convergences," 1970.

[3] J. Weiss, "The dynamics of enstrophy transfer in two-dimensional hydrodynamics," 1991.

[4] D. Kang and E. N. Curchitser, "Gulf Stream eddy characteristics in a high- resolution ocean model: Gulf Stream Eddy Characteristics," 2013.

[5] H. Toye, P. Zhan, G. Gopalakrishnan, A. Kartadikaria, H. Huang, O. Knio, and I. Hoteit, "Ensemble data assimilation in the Red Sea: sensitivity to ensemble selection and atmospheric forcing.," 2017.